Interdisziplinäres Praktikum
181.149

Alexander Ölzant
9301547
E 190 884 423

27. Januar 2007

Inhaltsverzeichnis

1 Szenario

Syndizierung einiger Genderdatenbanken im Informatik-Umfeld mittels Lixto und ‘ßu Fuss’”, Zusammenfassung von Inhalten in einem RSS-File zur Verfuegbarkeit in einem RSS-Reader.

Auflistung der verwendeten Datenbanken:

http://www.univie.ac.at/gender/	Genderreferat Universität Wien
http://www.gem.or.at/	Gender Mainstreaming im Europischen Sozialfonds (ESF): Mehrere Themenbereiche
http://www.genderandcomputing.no/	Weblog von Hilde Corneliussen (Department of Humanistic Informatics, Univ. of Bergen)
http://www.gendernet.at/	European Network for Assessment, Validation and Dissemination of Gender Mainstreaming Strategies in Vocational Guidance and Qualification (Methoden, Projekte)

2 Scraping mit `Lixto`

Entsprechend dem Tutorial wurden Bereiche der Ursprungsdokumente mit dem `Lixto Visual Wrapper` markiert und die Projektfiles mit den Wrappern zu den jeweiligen Source-Komponenten am Transformation Server eingebunden.

Als erster Ansatz wurden dazu feingranular den Quellen unterschiedliche XML-Tags zugeordnet, letztlich kam aber zur Vereinfachung des Syndizierungsprozesses eine Struktur zur Verwendung, bei der nur `items` im `rootPattern` definiert wurden, die jeweils wieder Muster mit den Bezeichnungen `title` und `description` beinhalteten (Abb. 1).

Am `Lixto Transformation Server` wurden anschliessend die Quellen mit einem `Integrator` vereint, wobei das Ausgabeschema von einem der Eingabeschemata uebernommen werden konnte. Dieses konnte fast direkt auch im Transformer iterativ wieder ausgegeben werden, wo aber noch die RSS-Tags im XSLT eingefügt wurden (s. Anhang A).

3 Websource-Eingabe/Verknuepfung mit Content Extractor

Die Wrapper-Dateien wurden dann zur Extrahierung der Rohdaten den Websources am `Lixto Transformation Server` zugeordnet, teilweise mit mehreren Quelldateien (Abb. 3). Komplizierte Navgationssequenzen waren

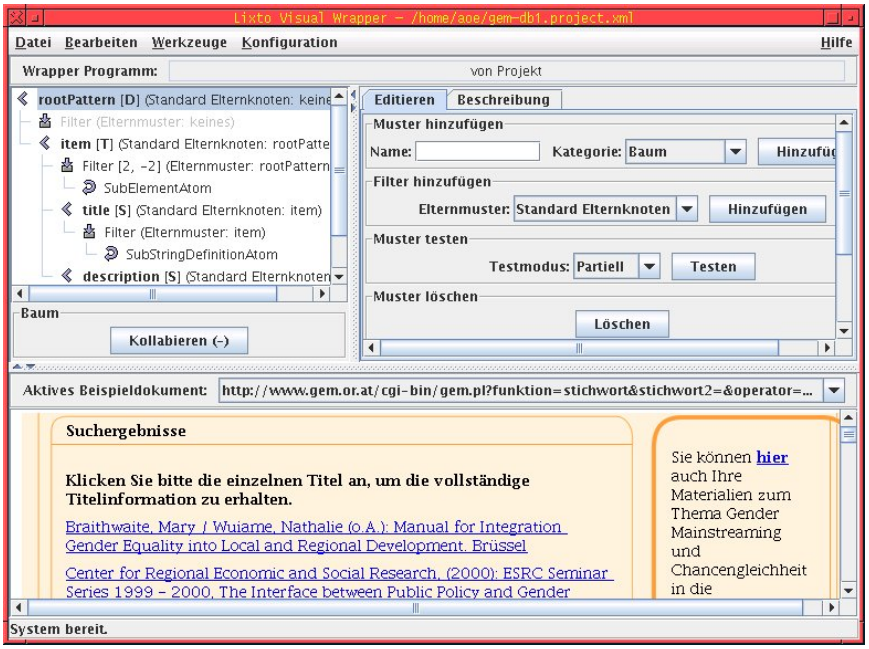


Abbildung 1: Wrapper-Darstellung einer der Komponenten

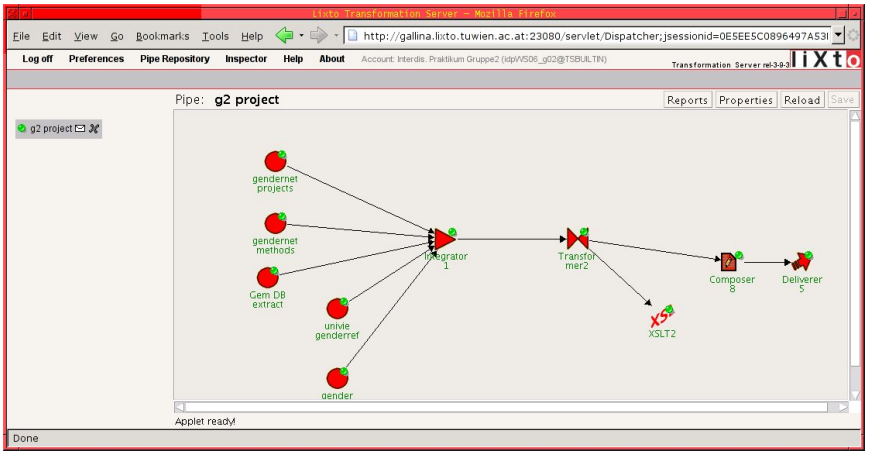


Abbildung 2: Übersicht über den Datenfluß am `Lixto Transformation Server`

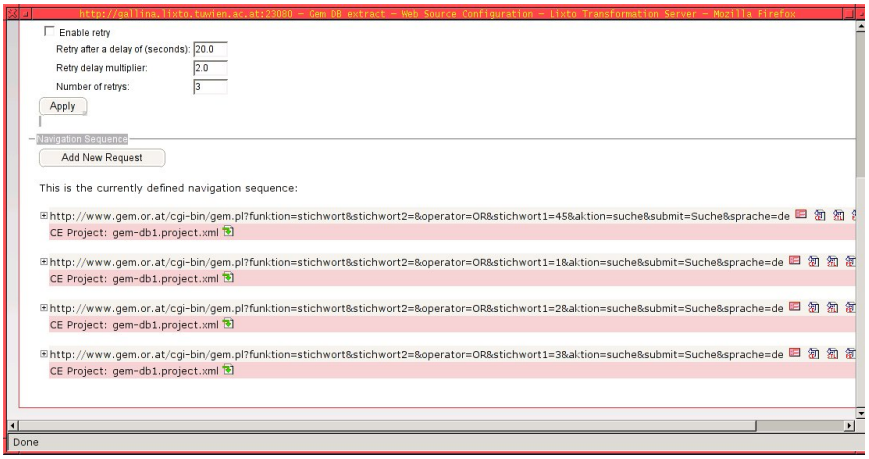


Abbildung 3: Websource (mehrere Quelldateien)

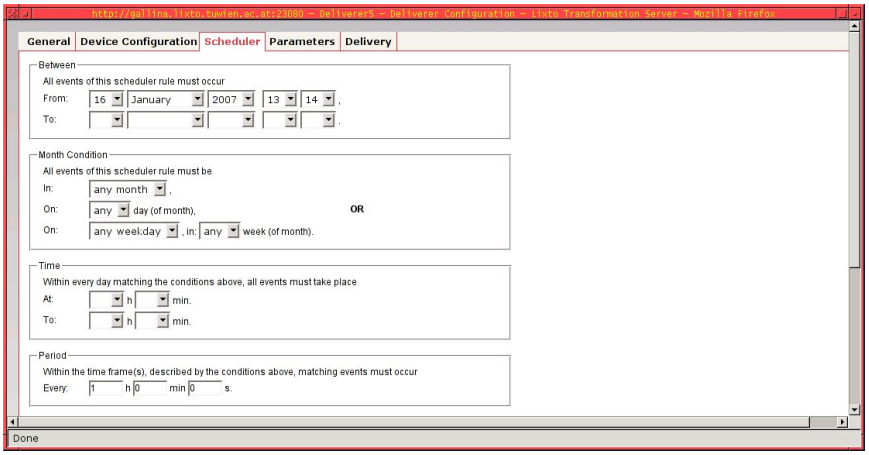


Abbildung 4: Scheduler: stündliche Aktualisierungen

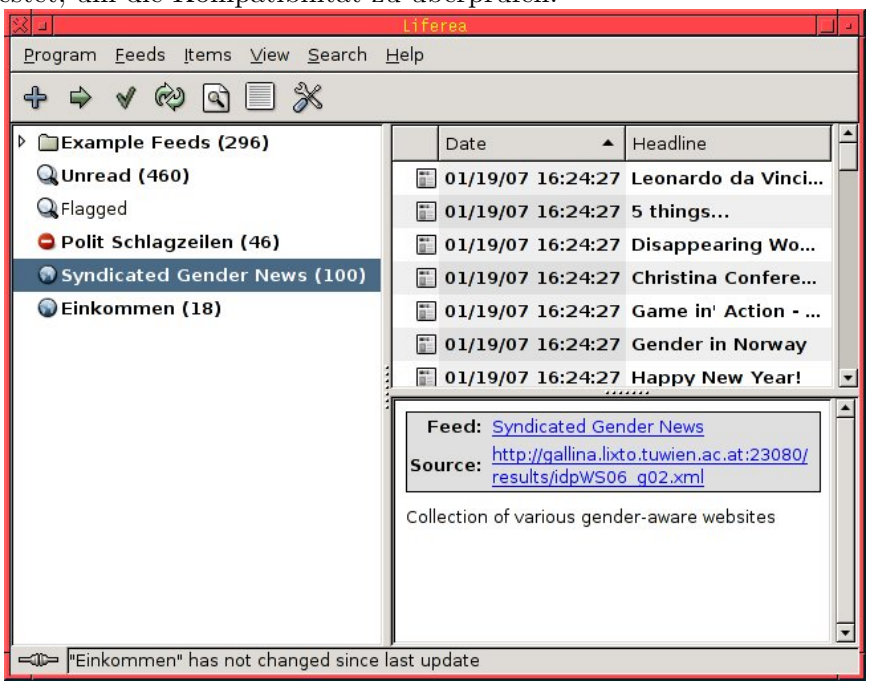
allerdings nicht notwendig, da keine Paginierung in den Ursprungsdokumenten ausgeführt war und die Quellen direkt (ohne POST von Forms) zugänglich sind.

4 Scheduler für den Deliverer

Um das RSS-File aktuell zu halten, wurde ein Scheduler eingerichtet, der die Pipe jede Studne ausführen sollte (Abb. 4). Allerdings wird eine delivery nur gemacht, wenn sich Datenbanken ändern.

5 Systemtest mit einem RSS-Reader

Das RSS 2.0 kompatible Resultat wurde dann mit einem Aggregator (liferea) getestet, um die Kompatibilität zu überprüfen.



A XSLT in der Transformer-Komponente

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:java="http://xml.apache.org/xslt/java"
                exclude-result-prefixes="java"
                version="1.0"
>
<!-- This file is automatically generated by the Transfomer Component. Do not edit!

<!-- variable to every inputnode -->
<!--  Input world id : world.idpWS06_g02_TSBUILTIN.p9.i36(Integrator1)-->
<xsl:variable name="INPUT_0" select="/Inputlist/Input0" />

<!-- config view - no selection is made,create only the join + outputstructure -->
<xsl:param name="TRANSFORMER_COND">0</xsl:param>

<xsl:variable name="SFDATE" select="java:java.text.SimpleDateFormat.new('MM/dd/yyyy
<xsl:variable name="SFTIME" select="java:java.text.SimpleDateFormat.new('kk:mm')"/>

<xsl:template match="/Inputlist">
<document>
<rss version="2.0">
<channel>
    <title>Syndicated Gender News</title>
    <link>http://gallina.lixto.tuwien.ac.at:23080/</link>
    <description>Collection of various gender-aware websites</description>


<xsl:for-each select="$INPUT_0/document/rootPattern/item">
<xsl:variable name="OUTPUT_1" select="./title" />
<xsl:variable name="OUTPUT_2" select="./description" />
<xsl:variable name="OUTPUT_3" select="./link" />

<!-- outputstructure -->
<item><title>  <xsl:value-of select="$OUTPUT_1"/>
</title>
<description>  <xsl:value-of select="$OUTPUT_2"/>
</description>
</item>

<!-- text contains enter -->
<xsl:text>&#xA;</xsl:text>
<!-- outputstructure end -->
</xsl:for-each>
```

```
</channel>
</rss>
</document>

</xsl:template>

<!-- Dismissing any unwanted text output -->
  <xsl:template match="text()"/>
</xsl:stylesheet>
```

B Alternativersion: Perl

RSS Output:
http://tigerente.htu.tuwien.ac.at/~aoe/studium/la_aoe/lixto/gender.rdf
Perl Source:
http://tigerente.htu.tuwien.ac.at/~aoe/studium/la_aoe/lixto/gender-rdf.pl

```
#!/usr/bin/perl
# make perl reasonably restrictive
use strict;
use warnings;

# parsers
use HTML::Form;
use HTML::TreeBuilder;
use CGI (qw/escape/);
# modules for outbound requests
require HTTP::Request;
use LWP::UserAgent;

############################################################
# start from there
my $base_uri="http://www.gem.or.at/cgi-bin/gem.pl?aktion=suche&" .
             "funktion=maske&sprache=de";

# for requests
my ($ua, $request, $response);
my (@forms, $form, %sc_urls);

############################################################
# make a request to collect the dropdown options
$request = HTTP::Request->new(GET => "$base_uri");
$ua = LWP::UserAgent->new;
$response = $ua->request($request);

# for testing: local version
#$response = cat("gem.html");
@forms = HTML::Form->parse($response, $base_uri);

# subroutine (see below) for extraction
my %out=listoptions($forms[0],"stichwort1");

############################################################
# q'n'd rss header ...
```

```
print <<"EOF";
<rss version="2.0">
EOF

my $escaped_uri=escape($base_uri);
for my $stichwort (keys %out) {
if ($stichwort) {
print <<"EOF";
<channel rdf:about="$base_uri">
<title>$out{$stichwort}</title>
<link>$escaped_uri</link>
EOF
print STDERR ">>>>>>>> $stichwort\n";
$forms[0]->param('stichwort1',$stichwort);
$ua = LWP::UserAgent->new;
$forms[0]->action =~ m#(.*/)#;
my $base_uri=$1;
# caching
my $content;
if (-f "gem-db-$stichwort.html") {
$content=cat("gem-db-$stichwort.html");
} else {
$response = $ua->request($forms[0]->click);
open O,">gem-db-$stichwort.html";
print O $response->content;
close O;
$content=$response->content;
}
my $tree=HTML::TreeBuilder->new_from_content($content);
for ([$tree->find('table')]->[3]->find('td')) {
my $link=$base_uri.[$_->extract_links]->[0][0][0];
if ($link =~ /gem.pl/) {
#my $content=join(" ",keys %$_);
# _parent _content align valign _tag
my $content=${$$_{_content}[0]}{"_content"}[0];
#>>>>>>>> 51
#Use of uninitialized value in concatenation (.) or string at
#   ./gender-rdf.pl line 61.
#Use of uninitialized value in concatenation (.) or string at
#   ./gender-rdf.pl line 61.
#Can't use string ("Es wurde kein Eintrag in der Dat") as a HASH ref
#   while "strict refs" in use at ./gender-rdf.pl line 64.

if ($content && $link) {
```

```
$escaped_uri=escape($link);
print <<"EOF";
<item>
<title>$content</title>
<link>$escaped_uri</link>
</item>
EOF
}
}
#print [$_->find("td")]->[0]->attr("text");
#print "   > ",$_->find("a")->attr("text"),"\n";
}

print <<"EOF";
</channel>
EOF

print STDERR "<<<<<<<< $stichwort\n";
}

}
############################################################
# q'n'd rss footer ...
print <<"EOF";
</rss>
EOF

sub cat {
my ($name, $oldifd, $data);
($name)=@_;
$oldifd=$/;
$/=undef;
open I,$name;
$data=<I>;
close I;
$/=$oldifd;
return $data;
}

sub listoptions ($$) {
my ($form,$name)=@_;
my %liste;

my @values=$form->find_input($name)->possible_values;
```

```
my @names=$form->find_input($name)->value_names;
for my $i (0..$#names) {
$liste{$values[$i]}=$names[$i];
}
return %liste;

}
```